Data cleaning using R

C. Tobin Magle, PhD

Based on

http://www.datacarpentry.org/OpenRefine-ecology-lesson/

and

https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf

Why is this useful?

- Data is rarely clean and tidy
  - Misspellings
  - White space
  - Multiple variables per column
  - Inconsistent coding
- Fixing it by hand takes forever
- Why not automate it with R?

Outline

- Cleaning data during import
  - read.csv() arguments
- Fixing imported data
  - Faceting and recoding
  - Data type conversion
  - Removing whitespace
  - Correcting misspellings
  - Splitting columns

Survey data

- **Rows**: observations of individual animals
- **Columns**: Variables that describe the animals
  - Species, sex, date, location, etc
- **Messy Data**
  - Misspellings
  - White space
  - Combined variables

Setup

- Download quickstart (http://bit.ly/2hvh54n)
- Open project in Rstudio by double-clicking the .Rproj file
- Look at the README file: contains information about variable names, column classes, and spellings of species names

Data cleaning with read.csv arguments

- **file** = “data/surveys_no_header.csv”)
- **header** = TRUE or FALSE
- **col.names** = c(…)
- **colClasses** = c(…)
- **na.string** = c(…)

Read in csv file

```
surveys <- read.csv(file = “data/surveys_no_header.csv”)
```

- What is wrong with the surveys data frame?

Specify no header

```
surveys<-read.csv(file = "data/surveys_no_header.csv",
                  header = FALSE)
```

- What’s wrong now?

Add the column names

```
surveys<-read.csv(file = "data/surveys_no_header.csv",
                  header = FALSE,
                  col.names = c("recordID", "mo", "dy", "yr",
                                "plot", "species", "scientificName",
                                "locality", "decimalLatitude",
                                "decimalLongitude", "county",
                                "state", "country", "sex", "hfl",
                                "wgt")
                  )
```

Data types

- Determines what you can do with it
- Numerical = math
- Categories = group + subset
- Text = human readable

read.csv() guesses the data type

R class	Description	What you can do?	How R guesses
int	Whole numbers	math	A column that’s all integers
num	Decimal numbers	math	A column that’s all numbers (integers or decimal)
factor	Integers with text labels	group by	Columns with any text (stringsAsFactors = TRUE)
chr	Plain text	text mining	Columns with any text (stringsAsFactors = FALSE)

Inspecting your data

- **str**() - all columns
- **class**() - one column
- **summary**() - summary stats
- Example: plot variable

```
> str(surveys)
'data.frame':   35549 obs. of  16 variables:
 $ recordID        : int  6545 5220 18932 20588 7020 7645 8641 9495 9583 98
 $ mo              : int  9 1 8 1 11 4 11 8 9 1 ...
 $ dy              : int  18 24 7 24 21 16 13 26 30 20 ...
 $ yr              : int  1982 1982 1991 1993 1982 1983 1983 1984 1984 1985
 $ plot            : int  13 20 19 12 24 24 15 9 15 13 ...
 $ species         : Factor w/ 48 levels "","AB","AH","AS",..: 2 2 4 4 3 3
 $ scientificName  : Factor w/ 29 levels "","  Amphispiza bilineata",..: 2
 $ locality        : Factor w/ 80 levels "","(Camporee Field) at Wallwood Be
cklawaha Arm of Lake Talquin, ca. 10 air mi S of Quincy.",..: 80 80 34 76 64
 $ decimalLatitude : num  NA NA NA 30.4 30.4 ...
 $ decimalLongitude: num  NA NA NA -84.2 -87.4 ...
 $ county          : Factor w/ 14 levels "","Escambia",..: 12 12 12 10 2 5
 $ state           : Factor w/ 9 levels "","Florida","Idaho",..: 5 5 5 2 2
 $ country         : Factor w/ 8 levels "","AUSTRALIA",..: 6 6 6 7 7 7 7 7
 $ sex             : Factor w/ 6 levels "","F","M","P",..: 1 1 1 1 1 1 1 1
 $ hfl             : int  NA NA NA NA NA NA NA NA NA NA ...
 $ wgt             : int  NA NA NA NA NA NA NA NA NA NA ...
```

```
> summary(surveys)
    recordID           mo               dy              yr
 Min.   :    1   Min.   : 1.000   Min.   : 1.00   Min.   :1977
 1st Qu.: 8888   1st Qu.: 4.000   1st Qu.: 9.00   1st Qu.:1984
 Median :17775   Median : 6.000   Median :16.00   Median :1990
 Mean   :17775   Mean   : 6.474   Mean   :16.11   Mean   :1990
 3rd Qu.:26662   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:1997
 Max.   :35549   Max.   :12.000   Max.   :31.00   Max.   :2002
```

Exercise 1:

- Open the readme file. Look at the column list
- Do you agree with the class designations?
- Does the structure of the surveys data frame match?

Specify column classes

```
surveys<-read.csv(file = "data/surveys_no_header.csv",
                  header = FALSE,
                  col.names = c("recordID", "mo", "dy", "yr",
                                "plot", "species", "scientificName",
                                "locality", "decimalLatitude",
                                "decimalLongitude", "county",
                                "state", "country", "sex", "hfl", "wgt")
                   colClasses = c("character", "factor", "factor", "factor", "factor",
                                  "factor", "factor", "character", "numeric", "numeric",
                                  "factor", "factor", "factor", "factor", "numeric",
                                  "numeric"))
```

Specify Missing data

- Specify missing data points (na.strings)
- NA – R’s standard for missing data
- Other common missing data indicators
  - “ “ - space
  - -999
  - “” - blank

Specify NA strings

```
surveys<-read.csv(file = "data/surveys_no_header.csv",
                  header = FALSE,
                  col.names = c("recordID", "mo", "dy", "yr",
                                "plot", "species", "scientificName",
                                "locality", "decimalLatitude",
                                "decimalLongitude", "county",
                                "state", "country", "sex", "hfl", "wgt")
                   colClasses = c("character", "factor", "factor", "factor", "factor",
                                  "factor", "factor", "character", "numeric",
                                  "numeric", "factor", "factor", "factor", "factor",
                                  "numeric", "numeric")
                    na.strings = c("NA", ""))
```

Data cleaning after import

- Type conversion
- Faceting
- Recoding
- Removing white space
- Splitting and combining columns
- Clustering to fix spelling errors

Data type conversions

- **as.character**() – input factor or numeric
- **as.numeric**() – input characters that can be interpreted as numbers; be careful with factors!
- **as.factor**() – character or numeric

Factor to character

```
class(surveys$plot)    #the plot variable is a factor

surveys$plot<-as.character(surveys$plot) #factor to character

class(surveys$plot)
```

Character to numeric

- **class**(surveys**$**plot)
- **surveys$plot<-as.numeric**(surveys**$**plot) #character to numeric
- **class**(surveys**$**plot)

Numeric to factor

- **class**(surveys**$**plot)
- **surveys$plot<-as.factor**(surveys**$**plot) #character to factor
- **class**(surveys**$**plot)

Exercise 2: Factor to numeric

- Year is currently stored as factor. Try:

```
year <- as.numeric(surveys$yr)
```

- What went wrong?

- How would you do it so that we get number conversions of the label?

Faceting with factor levels

- List factor levels: **levels**()

```
sex<-surveys$sex
levels(sex)
nlevels(sex)
summary(sex)
```

Exercise 3: levels

1. Using levels, find out **how many years** are represented in the census.

1. Which years have the **most and least observations**?

Recoding variables

- The **recode**() function is defined in the dplyr package
  - Input = a factor, <old level> = <new level>, …
  - Output = a new factor

```
library(dplyr)
sex<-recode(sex, “P” = “other”, "R"="other", "Z" = "other")
summary(sex)
```

Exercise 4

- Two of the scientific names have a strange symbol instead of a space: "Dipodomys\xe6sp." and "Onychomys\xe6sp”.

- Use the recoding techniques we just learned to fix this error

Dealing with whitespace

- **trimws**() – removes leading and trailing white space
  - Input – character or factor
  - Output- factor

```
# Trim whitespace
surveys$scientificName<- trimws(surveys$scientificName)

# Type conversion back to factor
surveys$scientificName<- as.factor(surveys$scientificName)
```

Fix misspellings with stringdist

```
library(stringdist) # load the stringdist library

stringdist("abc", "abc") #no difference = 0 distance
stringdist("abc", "abd") #1 difference = distance of 1
stringdist("abc", "cba") #2 differences = distance of 2
stringdist("abc", "def") #3 differences = max distance of 3
```

Compare scientificNames

```
sp_names<-surveys$scientificName

stringdist(spnames,
                "Ammospermophilus harrisii")

levels(sp_names)
```

Specify correct spellings

```
codes<-c("Ammodramus savannarum",   "Ammospermophilus harrisii",
         "Amphispiza bilineata",          "Amphispiza cilineata",
         "Baiomys taylori",               "Calamospiza melanocorys",
         "Callipepla squamata",           "Campylorhynchus brunneicapillus",
         "Chaetodipus baileyi",           "Cnemidophorus tigris",
         "Cnemidophorus uniparens",   "Crotalus scutalatus",
         "Crotalus viridis",              "Dipodomys merriami",
         "Dipodomys ordii",               "Dipodomys spectabilis",
         "Dipodomys sp.",                 "Onychomys leucogaster",
         "Onychomys torridus",            "Onychomys sp.")
```

Approximate string matching

- **amatch**() – matches strings to a list of accepted values
  - input – a sequence of strings and a sequence of acceptable values
  - Output – a sequence of numbers matching position in table
    - No match = NA

```
#create a list of which names match each codes
i<-amatch(x = sp_names,       #the list of things you want to code
                table = codes)       #the list of acceptable values
```

Quality control

```
#data frame with columns for raw text and the assigned code
sp_names_df<-data.frame(rawtext = sp_names,
                        code = codes[i])
```

Look at sp_names_df: do you see NAs?

Method name	Description
osa	Optimal string aligment, (restricted Damerau-Levenshtein distance).
lv	Levenshtein distance (as in R's native adist).
dl	Full Damerau-Levenshtein distance.
hamming	Hamming distance (a and b must have same nr of characters).
lcs	Longest common substring distance.
qgram	*q*-gram distance.
cosine	cosine distance between *q*-gram profiles
jaccard	Jaccard distance between *q*-gram profiles
jw	Jaro, or Jaro-Winker distance.
soundex	Distance based on soundex encoding (see below)

For more information see stringdist-metrics documentation

Specify matching method

```
i<-amatch(sp_names, codes,
                  method = "cosine")

#create comparison df again
sp_names_df<-data.frame(rawtext = sp_names,
                                    code = codes[i])
```

Look at sp_names_df: do you see NAs?

More QC

```
#Are there any unassigned? - not at the top
sum(is.na(sp_names_df$code))

#is this the same as the original dataset?
sum(is.na(sp_names_df$code) ==
sum(is.na(surveys$scientificName))
```

Splitting columns using separate()

- **separate**() – turns a single character column into multiple columns

- Found in the tidyr package

```
surveys<-separate(data = surveys,                    #your data frame
                  col = scientificName,              #column to split
                  sep = c(" "),                      #what to split on
                  into = c("genus", "sp"), #names of new columns
                  remove = FALSE                     #keeps original column
                 )
```

Need help?

- Email: tobin.magle@colostate.edu

- Data Management Services website: http://lib.colostate.edu/services/data-management

- Data Carpentry: http://www.datacarpentry.org/
  - R Ecology Lesson: http://www.datacarpentry.org/OpenRefine-ecology-lesson/
- Data cleaning reference :
  - https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf